APPENDIX B

SOFTWARE DESIGN DESCRIPTION FOR THE ARABIC NAME SEARCH ALGORITHM FOR CLASS - E (ANA - E)

TABLE OF CONTENTS

SOFTWARE DESIGN DESCRIPTION FOR THE ARABIC NAME SEARCH ALGORITHM FOR CLASS - E (ANA - E)

1. INTRODUCTION

1.1. Purpose

The variation that can occur in the transcription of Arabic names into roman representation poses formidable problems for retrieval systems with very large databases that depend solely on standard string-comparison techniques. For example, the following names are transcription variants of the same name: **SALEHUDDINE, IMHEMED** and **SAALAH EL DEEN, MUHAMMED**. The significant differences in their spellings and in the distribution of white space would virtually preclude any possibility of identifying these names as similar enough to be candidates for retrieval if the usual techniques were applied. The task, then, is to capture the relatedness of these names and to incorporate the insights into their commonality into the retrieval system.

Arabic names are made up of a Given Name (GN) (usually one, although compound names may occur: **SAMIR; MOHAMAD ALI**) and a string of familial (paternal) relations following the GN (**ABD EL KADEER SAMIR ABD EL LATIF**). The string following the GN is generally made up of GNs which are taken from the father, grandfather and other relations. Only in rare cases can any of these segments be identified as a SN, i.e., a name used by every member of the family to signal family membership. The full string following the GN provides crucial information about the individual that is lost if it is sometimes in the GN field and sometimes in the SN field. So, positioning names that occur after the first GN in the SN field provides the opportunity for better matches.

1.2. Scope

In 1996, LAS proposed an initial solution to the problem, the salient feature of which was to level spelling differences and thereby generate one representation for the myriad spellings of a single name. This process is known as ***regularization***, a technique that was implemented in the Legacy CLASS system as Legacy ANA. Legacy ANA is a preprocessing module that feeds into the Legacy CLASS search system. The general characteristics of Legacy ANA are:

1) Both query and add procedures are identical for the Legacy ANA system.

2) A rudimentary Arabic name identifier (ANI) determines if an input name qualifies for handling by the Legacy ANA algorithm. All names that qualify for Legacy ANA handling are also sent to the generic processing module provided by Legacy CLASS and to the DOB processor, when appropriate.
3) A set of regularization rules is applied to the Arabic input name, leveling the spelling differences of the name segments to the most common representation of the input name. **IMHEMED** and **MUHAMMED** will both be regularized to the form **MUHAMAD**, for example. Some title/affix/qualifier information may also be removed from the name to focus on the name stem. The regularization rules are rewrite rules that use notation developed solely for this processor. The rule engine necessary for implementation of the regularization rules was also developed specifically to handle the Legacy ANA regularization rules. The output of the regularization component is a regularized form of the input name.
4) The output of the regularization component (the regularized form of the name) serves as the input to the generic CLASS search system. CLASS produces standard compressed-name keys, but on the regularized form that is the output of Legacy ANA. CLASS accesses the database records through the keys on the regularized form. (The keys are generated for both queries and adds and are stored with the record when a record is added to the database.)
5) A digraph match is then performed on the regularized record and query forms to determine name similarity. The match criteria are those of CLASS.

The ANA-E system is an enhancement of the Arabic name search system that was developed by LAS for the Legacy CLASS system. The principle of name regularization remains the same in ANA-E, although the design and approach of ANA-E are different in a number of important ways.

1) An independent Name Classifier (ANC-E) has been developed. (The ANC-E design description is provided as Attachment A in LAS Linguistic Memo CT970044, May 30, 1997) ANC-E will direct input names to the Arabic and/or Hispanic processors. Its functionality is far more sophisticated than the Arabic name typer (ANI). All records will also be directed to the CN pipe of CLASS-E and to the DOB pipe, if appropriate.
2) A significant amount of preprocessing of the name takes place in ANA-E that recognizes the unique character of Arabic names, focusing on the leftmost GN as the most stable element in the name and rejoining all other GN segments with their SN partners.
3) The regularization rules and rule engine have changed. The rules are represented in standard regular expressions and the format has changed. The rule engine uses different match techniques, is much simpler in its implementation and therefore can be easily applied to other rule sets.
4) The output of the regularization rules is a computationally viable form, one that may not be the most common representation of a name (as was a requirement of Legacy ANA).

5) The new ANA-E rule language has already allowed a dramatic increase in the amount of regularization that takes place. For those records that qualify as Arabic, a decrease of 13% in the number of different regularized forms from Legacy ANA (as of March 1997) to the proposed ANA-E regularization rules demonstrates the greater flexibility of the ANA-E language. The system must therefore accommodate fewer distinct name segments, reducing processing time and increasing the scope of retrieval.
6) Arabic-specific keys have been generated to account for the nature of Arabic names and at the same time permit some unpredictable variation. The regularization rules account for much of the predictable variation in names but are only incidentally able to accommodate unpredictable variation (e.g., error).
7) Retrieval is based on keys that represent a class of pre-determined variants of a name segment and are formed from the GN1 and SN segments.
8) Gender has been added as a search criterion to reduce the occurrence of crossed-gender retrievals.
9) The filtering techniques used in ANA-E demonstrate much greater granularity and sensitivity to Arabic-specific name characteristics.

The CLASS-E system will support several concurrent record search processes. The Multi-Pipe Architecture (MPA) already in place supports the generic-CLASS search process and a distinct Date-of-Birth process. (Because the Legacy ANA system serves as a preprocessing module that feeds into the generic-CLASS processing pipe, Legacy ANA is not characterized as a separate search pipe.)

In CLASS-E the Multi-Pipe Architecture will be extended to include a distinct Arabic processing pipe, a distinct Hispanic processing pipe as well as perhaps others in the future. An input name may be submitted to more than one processing pipe. It is a business decision of CA/EX/CSD to determine to which and how many pipes to pass a given input name. It is suggested that names classified as Arabic by the Advanced Name Classifier for CLASS-E (ANC-E) be submitted to multiple processors: the generic CLASS-E processing pipe, the DOB processing pipe and the ANA-E processing pipe.

1.3. Definitions and Acronyms

ACOB	Arabic COB Category Data Store
ADE	Arabic Data Evaluator
AFS	Arabic Filter and Sorter
AG	Applicant Gender (user supplied)
AGI	Arabic Gender Identifier
AKG	Arabic Key Generator
ANA - E	Arabic Name Search Algorithm for CLASS - E
ANC - E	Advanced Name Classifier for CLASS - E
ANI	Arabic Name Identifier (Legacy ANA)
ANR	Arabic Name Regularizer
ANT	Arabic Name Type Data Store
APP	Arabic Pre-Processor
ARE	Arabic Rule Engine

ARR	Arabic Regularization Rules Data Store
ASE	Arabic Search Engine
ASP	Arabic Segment Positioner
ATD	Arabic TAQ Data Store
ATP	Arabic TAQ Processor
COBPROX	COB Proximity Data Store
DELETE	Segment will be removed from any further consideration in the name matching process; it will contribute marginally to the filtering process. The segment will *not* be removed from the record.
DISREGARD	Segment will be removed from consideration in the name retrieval process but will contribute to the filtering and sorting processes.
DI_VAL	Digraph Value
F	Female Gender
FNU	First Name Unknown
FP	Filter Parameter Data Store
GN	Given Name
GN1	Leftmost GN1 segment
GNDR	Gender
GNTHR	Given Name Threshold (Filter)
GN_VAL	Final Given Name Value
Given Name Field	All name segments to the right of the comma
HF	High Frequency
HFI	Arabic High Frequency Name Identifier
HK	High Frequency Key
HS	High Frequency Search Key
INITGN	Given Name Initial Value
INITSN	Surname Initial Value
K-Key	Special Key formed to handle name segments with "k"
Legacy ANA	Arabic Name Algorithm for Legacy CLASS
LF	Low Frequency
LTF	Linguistic Trace Facility
M	Male Gender
OPSN	Out-of-Position Surname Segment
OPVAL	Out-of-Position Value (Filter)
PK	Primary Key
RCL	Refusal Code Level Data Store
Record Gender	Gender determined for a record based on two gender validators, input gender and HF name gender; all gender indicators must agree.
Regularization	Rule-based process that levels the differences among the roman spellings of a single Arabic name
RG	Record Gender
RLYOB	Refusal Code Level/Year-of-Birth Range Data Store
Segment	Any single name piece, surrounded by white space
SI	Single-Part Key
SK	Search Key
SNTHR	Surname Threshold (Filter)
SN_VAL	Final Surname Value
SP	Special Key
SS	Standard Search Key
Surname Field	All name segments to the left of the comma
TF	TAQ Filter Data Store

TAQ	Title/Affix/Qualifier
TAQAGN	Value for Missing TAQ in the Given Name
TAQASN	Value for Missing TAQ in the Surname
TAQXGN	Value for TAQ DELETE in Given Name
TAQXSN	Value for TAQ DELECT in Surname
U	Unknown (Ambiguous) Gender
WK	Wild-Card Key
YR	Year-of-Birth Range Data Store

2. MODULE DECOMPOSITION

2.1. The Arabic Name Search Algorithm for CLASS-E (ANA-E) will consist of three primary components (see pages 6-9 for graphic representations of these components):

- the Arabic Pre-Processor (APP),
- the Arabic Search Engine (ASE), and
- the Arabic Filter and Sorter (AFS).

2.2. ARABIC PRE-PROCESSOR MODULE DECOMPOSITION

2.2.1. **Identification**

This module is known as the Arabic Pre-Processor (APP).

2.2.2. **Type**

The APP is the first programming module in the larger ANA-E algorithm and consists of subordinate functions that manipulate the name segments in various ways to prepare the name for creation of search keys by Arabic Search Engine.

2.2.3. **Purpose**

Because of the significant variation that can occur in names that have been romanized from the original Arabic script, Arabic names will benefit from attempts to level the spelling differences. In addition, the standard format of an Arabic name is Given Name followed by a string of segments that indicate familial relations. In many countries, none of these segments functions as what is standardly referred to as a *surname*. What is determined to be a Surname for purposes of a CLASS retrieval (i.e., what is placed in the Surname field) is therefore quite arbitrary. Arabic names will consequently benefit from movement of name segments that would contribute to a potential match.

2.2.4. **Function**

2.2.4.1. The Arabic Pre-Processor (APP) will accept as input any name that has been identified as Arabic by the Advanced Name Classifier for CLASS-E (ANC-E) and will prepare a name for the Arabic Search Engine by applying Arabic regularization rules to the name segments and reorganizing the name according to Arabic naming principles.

2.2.4.2. The APP can alternatively create a name object that "knows" characteristics about itself and collects information as it proceeds through the processing functions.

2.2.5. **Subordinates**

- Arabic Name Regularizer (ANR)
- Arabic TAQ Processor (ATP)
- Arabic Data Evaluator (ADE)
- Arabic Segment Positioner (ASP)
- Arabic Gender Identifier (AGI)

2.3. ARABIC NAME REGULARIZER MODULE DECOMPOSITION

2.3.1. **Identification**

This module will be known as the Arabic Name Regularizer (ANR) and will consist of one subordinate processor, the Arabic Rule Engine, which will access and apply the rules in one data store, the Arabic Regularization Rules (ARR) Data Store.

2.3.2. Type

The ANR is a program that

- will operate on a full surname string and a full given name string of an add or query record,
- will generate a regularized form for each name segment or string of name segments to which the regularization rules have applied, and
- will submit the regularized form to other functions in the APP to continue to prepare the name for submission to the Arabic Search Engine.

2.3.3. Purpose

The transcription of Arabic names from their native format (Arabic script) to the roman alphabet is highly variable; few, if any, transcription standards exist. Such rampant variation poses significant problems for string matching and retrieval systems; there are often too many characters that differ to effect a retrieval in character-based retrieval systems. For example, **MUHAMMAD** and **IMHEMED** are roman spellings of the same name; that is, they are represented by the same string of characters in the Arabic script. Leveling the differences in roman spelling, wherever possible, would improve record retrieval dramatically.

2.3.4. Function

The ANR applies a set of regularization rules (ARR) to the surname and to the given name through the Arabic Rule Engine and produces a regularized form for any name segment or string of segments to which the rules can apply.

2.3.5. Subordinates

The ANR consists of one subordinate function, the Arabic Rule Engine, which accesses the Arabic Regularization Rule Data Store.

2.4. ARABIC RULE ENGINE MODULE DECOMPOSITION

2.4.1. Identification

This function is known as the Arabic Rule Engine (ARE).

2.4.2. Type

The ARE is a program that attempts to apply transformation rules to an input string of characters and to effect a change in that string.

2.4.3. **Purpose**

2.4.3.1. The development of rules that are implemented in standard and readily accessible regular expressions allows for use of a less idiosyncratic rule engine than the one developed for Legacy ANA. (See Arabic Regularization Rule Data Store, Section 3.3).

2.4.3.2. The Arabic Regularization Rules (ARR) require an implementation module to effect the changes specified in the rules. The ARE plays that role.

2.4.3.3. The ARE replaces the rule engine developed for Legacy ANA. It is simpler and more generic and can be used for other rule implementations.

2.4.3.4. The ARR are more easily altered and reviewed.

2.4.4. **Function**

2.4.4.1. The ARE accepts a full surname (SN) or full given name (GN) string as input.

2.4.4.2. The ARE will add a white space to the beginning of the SN or GN string that it accepts to serve as boundary markers.

2.4.4.3. The ARE scans the input string from left to right and attempts to match the Match Context of a rule.

2.4.4.4. If the ARE is able to identify a Match Context, it checks to see if the Pre- and Post-Contexts specified in the rule are present.

2.4.4.4.1. If the Pre- and Post-Contexts specified in the rule match, then the ARE applies the rule and makes the specified change in the Match Context, producing the Output.

2.4.4.4.2. The ARE then returns to the top of the rule set and attempts to identify a Match Context beginning with the character immediately following the previous Match Context.

2.4.4.5. If no match is found, the ARE moves to the Match Context of the next rule.

2.4.4.6. If no rule has fired, the default rule applies: the character output is the character itself. E.g., S → S

2.4.4.7. **Arabic Regularization Rules (ARR)**

2.4.4.7.1. The ARRs are written as regular expressions and use, for the most part, regular expression notation. See Section 3.3 for ARR details.

2.4.4.7.2. The ARRs use defined metasymbols.

2.4.4.7.3. The ARE must be able to recognize all regular expression notation and metasymbols of the ARR and implement them.

2.4.4.7.4. ARRs have the format:

Figure 1: Format: Arabic Regularization Rule

PRE-CONTEXT	MATCH CONTEXT	POST-CONTEXT	→	OUTPUT

2.4.4.7.5. Rule Ordering

2.4.4.7.5.1. Rule ordering is important because the **first** rule for which the ARE finds a Match Context (and the Pre- and Post-Contexts match as well) will apply. Once the rule has applied to the Match Context, no other rules will apply to it: *No* following rule will then fire on that Match Context.

2.4.4.7.5.1.1. The rules must have internal ordering based on the Match Context only.

2.4.4.7.5.1.2. Rules may intrude in the ordering of the Match Context if they are applicable to another phenomenon.

2.4.4.7.5.1.2.1. For example, an MI → NE rule will need to precede an M → N rule or the MI → NE will never apply.

2.4.4.7.5.1.2.2. A rule that applies to a Match Context where A → AW could intervene between the "M" rules and have no effect on the order of application of the "M" rules.

2.4.4.7.5.1.2.3. In general, rules with longer character strings in the Match Context need to precede rules with shorter character strings.

2.4.4.7.5.1.2.4. Care must be taken when rules have symbols for optional characters, for example. The ordering of an M?L rule (a rule that can apply to ML or L) must be carefully placed with respect to other rules that apply to M and L.

2.4.4.7.5.2. The Output of one rule does *not* form the input to another rule.

2.4.4.7.5.2.1. Only one rule applies to a character or character string that matches a Match Context.

2.4.4.7.5.2.2. The first rule that matches all three contexts in the Match, Pre- and Post- Context order is applied.

2.4.4.7.5.2.3. The Output of a rule cannot then be changed.

2.4.4.7.5.2.4. Rules must be written so that they stand alone: rules are not interdependent.

2.4.4.7.5.3. If the ARE is able to match the Match Context, the Pre- and Post-Contexts are examined.

2.4.4.7.5.3.1. If the Pre- and Post-Contexts both match, the ARE effects the change in the Match Context indicated in the OUTPUT.

2.4.4.7.5.3.1.1. The next available context in the input string to be considered for a match immediately follows the previous *Match Context.*

2.4.4.7.5.3.1.2. For example, if HEIMER is the input string and a rule applies to HEIM to make it GIM, the next available context for consideration is the E of ER (following HEIM). If an E rule is to apply, it can only apply to the *second* E, not that of the previous Match Context (H*E*IM).

2.4.4.7.5.3.1.3. The Output of a previous rule cannot be the Pre- or Post-Context of a following rule.

2.4.4.7.5.3.2. The rule is applied only if the ARE is successful in matching the Match Context and the Pre- and Post-Contexts.

2.4.4.7.6. There is no backtracking in the ARE.

2.4.4.7.7. The output of successful application of rule(s) by the ARE is a regularized Arabic form. The output of the ARE can be in any string form (e.g., binary, regular expression, characters).

2.4.4.8. **Subordinates**

None.

2.5. ARABIC TAQ PROCESSOR MODULE DECOMPOSITION

2.5.1. **Identification**

This function is known as the Arabic TAQ Processor (ATP).

2.5.2. **Type**

The ATP is a function that identifies titles (T), affixes (A) and qualifiers (Q), as specified in the Arabic TAQ Data Store, and implements the disposition indicated in that table.

2.5.3. **Purpose**

Arabic names frequently contain peripheral name elements, such as ABDEL, ABU, AL. Matching on these segments is not generally useful; the name segments with information value are the name stems, RAHMAN, SAYED, HANAWI. Removal of or disregard for the peripheral name elements allows more emphasis to be placed on the name stems.

2.5.4. **Function**

2.5.4.1. The ATP will access the Arabic TAQ Data Store (ATD) to identify titles (e.g., USTAAZ), affixes (e.g., EL DIN) and qualifiers (Q) that occur in the regularized name.

2.5.4.2. The ATP will tag as a T, P, I, S, or Q any such segments found in the name, as specified in the ATD.

2.5.4.2.1. The ATP will scan the full SN or GN field for any TAQ segments.

2.5.4.2.2. If the ATP identifies a segment, it will tag the segment with the ID_NO and disposition, as indicated in the ATD.

2.5.4.2.3. If the following segment is also a TAQ segment, it will tag the segment with the ID_NO and disposition, as indicated in the ATD.

2.5.4.2.4. This will continue until all *consecutive* TAQ segments have been tagged.

2.5.4.2.5. When the ATP encounters a following segment that is not a TAQ segment, it will treat that segment as a stem.

2.5.4.2.5.1. Each TAQ segment identified up to that point will be given the TAQ_TYPE P (prefix) and each will be associated and stored with the following stem.

2.5.4.2.6. The ATP will move to the next segment following the stem and will repeat the TAQ identification process.

2.5.4.2.6.1. The ATP will tag all TAQ segments with the ID_NO and disposition.

2.5.4.2.6.2. When the ATP encounters a stem, it will tag each TAQ segment (not yet associated with a stem) with the TAQ_TYPE P and will associate and store each TAQ segment with the following stem.

2.5.4.2.7. If the ATP encounters a TAQ segment (or segments) that has no following stem, it will access the ATD to determine if the TAQ type is a Suffix (S).

2.5.4.2.7.1. If the TAQ has a TAQ_TYPE S, the TAQ will be associated and stored with the *preceding* stem.

2.5.4.2.7.2. The preceding stem may already have prefixal TAQs.

2.5.4.2.7.3. If the TAQ type is not equal to S, the TAQ will be tagged a Stranded Affix.

2.5.4.3. The ATP will process any TAQ segments identified according to the treatment indicated in the ATD. (See Section 3.4.)

2.5.4.3.1. Treatment options include DELETE and DISREGARD.

2.5.4.3.2. **DELETE** means that the segment is completely disregarded in the remainder of the name search process and contributes marginal information to the filtering process. (N.B. The segment is not deleted from the record.)

2.5.4.3.3. **DISREGARD** means that the segment is disregarded in the remainder of the name search process but contributes to the evaluation of the name in the filtering processes.

2.5.4.4. TAQ Tag

2.5.4.4.1. The TAQ tag will reference the ID_NO of the TAQ.

2.5.4.4.2. The TAQ tag will reference the indicated treatment of the TAQ segment.

2.5.4.4.3. The TAQ tag will be associated with a name stem, unless marked as a Stranded Affix.

2.5.4.4.4. Surnames containing the prefix AL (e.g., AL IDRISI) will be specially marked.

2.5.4.5. The TAQ tag will assist in the sorting of records (see Section 2.12, the Arabic Filter and Sorter (AFS)).

2.5.5. **Subordinates**

None.

2.6. ARABIC DATA EVALUATOR MODULE DECOMPOSITION

2.6.1. **Identification**

This function is known as the Arabic Data Evaluator (ADE).

2.6.2. Type

The ADE is a function that "corrects" data entry errors by generating one or more alias records.

2.6.3. Purpose

Arabic names are conventionally a Given Name followed by a string of (usually paternal) relationships, elements of which are routinely deleted. Some data entry operators have apparently attempted to capture the fact that the Arabic name is closer to a single name string by entering XXX into the Given Name field, cf., presumably the XXX permitted in the COB or DOB fields. Because XXX is not a conventional representation of any Given Name information, it interferes with the name search and will be altered.

2.6.4. Function

The ADE will determine if the leftmost Given Name segment (only) is XXX. If so, it will change that string to FNU and generate an alias add record or query.

2.6.5. Subordinates.

None.

2.7. ARABIC SEGMENT POSITIONER MODULE DECOMPOSITION

2.7.1. Identification

This function is known as the Arabic Segment Positioner (ASP).

2.7.2. Type

The ASP is a processing module that operates on the preprocessed, regularized name and moves name segments from the Given Name field into the Surname field. Alias records will be produced to reflect format changes.

2.7.3. Purpose

Arabic names are made up of a Given Name (GN) (usually one, although compound names may occur: **SAMIR; MUHAMAD ALI**) and a string of familial (paternal) relations following the GN (**ABD EL KADEER SAMIR ABD EL LATIF**). This string is generally made up of GNs which are taken from the father, grandfather and other relations. In most cases, none of these segments be identified as a Surname, i.e., a name used by every member of the family to signal family membership. The full string following the GN provides crucial information about the individual that is lost if it is sometimes in the GN field and sometimes in the SN field. So, positioning names that occur after the first GN in the SN field provides the opportunity for better matches.

2.7.4. **Function**

2.7.4.1. The ASP will move all segments to the right of the leftmost GN (GN1) (in the preprocessed, regularized name) to the leftmost SN position, preserving the order of the moved segments.

Figure 2: Movement of Segments into SN Field

FARUK, MUHAMAD SAMIR ABDULA	→	SAMIR ABDULA FARUK, MUHAMAD

2.7.4.2. The leftmost Given Name (GN1) segment will not be moved into the Surname field *except*

a) if there is one and only one GN segment and
b) if there is one and only one SN segment which has been tagged as having the prefix AL,
c) then the ASP will generate an alias record with the SN and GN inverted.

Figure 3: Inversion of SN and GN with AL in the SN Field

SURNAME	GN1	
(AL) IDRISI	YUSEF	→
YUSEF	(AL) IDRISI	

2.7.4.3. The GN1 may be a name segment, an initial, or FNU. (See Section 2.9, the Arabic Search Engine (ASE) for additional information.)

2.7.5. **Subordinates**

None.

2.8. ARABIC GENDER IDENTIFIER MODULE DECOMPOSITION

2.8.1. **Identification**

This function is known as the Arabic Gender Identifier (AGI).

2.8.2. **Type**

2.8.2.1. The AGI is a function that will apply after the ANR has produced a regularized representation of the input name and the Arabic Segment Positioner (ASP) has moved all GN segments other than the GN1 into the SN field.

2.8.2.2. For the AGI to derive record gender, the data input operator will need to supply gender for each record added to the database and for each query during the data entry process.

2.8.3. **Purpose**

2.8.3.1. Crossed-gender records are of little value to the system user.

2.8.3.2. Arabic gender is reliably predictable from the GN1.

2.8.3.3. Records that have crossed gender will receive lowered match values during the filtering and sorting process.

2.8.3.4. Record gender requires gender validation from two sources: gender received during the data entry process and predictable gender associated with Arabic names.

2.8.3.5. Record gender reduces the chance of associating gender with a name that may be misspelled.

2.8.4. **Function**

2.8.4.1. The AGI will derive the record gender for all record adds and queries.

2.8.4.1.1. For each query and add name, the AGI will derive record gender from user-supplied gender input and from the gender that has been assigned to the GN1.

2.8.4.1.2. A *minimum* of two gender indicators is required for a gender assignment of M or F.

2.8.4.2. For record adds, gender received as input from the data entry process will be stored with the record.

2.8.4.3. For record queries, the user will input the gender of the applicant at query time.

2.8.4.4. For both adds and queries, the AGI will access the Arabic Name Type Data Store (ANT) and will assign the gender value to the GN1 segment, as indicated in the ANT (GENDER). (See Section 3.5.)

2.8.4.4.1. If the name is present in the ANT, the gender associated with the name segment will be compared to the data entry gender.

2.8.4.4.1.1. If the gender indicators match, the matching value will become the record gender.

2.8.4.4.1.2. If the gender indicators do not match, the record gender will be Unknown (U).

2.8.4.4.2. If the name is not present in the ANT, the record gender will be marked as Unknown (U).

2.8.5. **Subordinates**

None.

2.9. ARABIC SEARCH ENGINE MODULE DECOMPOSITION

2.9.1. **Identification**

This module is known as the Arabic Search Engine (ASE).

2.9.2. **Type**

The ASE is a processing module that accepts the output of the Arabic Preprocessor (APP), generates retrieval keys through the Arabic Key Generator, retrieves candidate records from the database based on the keys and submits those candidate records to the Arabic Filter and Sorter (AFS) Module.

2.9.3. **Purpose**

The regularized, repositioned names generated by the APP will be, in general, a representation of the canonical form of the Arabic name. The search process will benefit from focus on the canonical form of the Arabic name.

2.9.4. **Function**

The ASE will retrieve records from the database whose stored keys match the keys generated for the query record.

2.9.5. **Subordinates**

The ASE has one subordinate module:

- Arabic Key Generator

2.10. ARABIC KEY GENERATOR MODULE DECOMPOSITION

2.10.1. **Identification**

This function is known as the Arabic Key Generator (AKG).

2.10.2. **Type**

The AKG is a function that will form keys from the GN1 and each SN segment of the preprocessed, regularized name for both record adds and queries.

2.10.3. **Purpose**

In order to reduce the number of records that must be compared by the Arabic Filter and Sorter Module, it is desirable to subset the Arabic database. (About 500,000 records have qualified as Arabic through the ANI name typing process and it is assumed that this number will continue to represent the approximate size of an Arabic database.) One mechanism for achieving a

subset is to generate keys for the input name. The Arabic keys are motivated by the nature of the Arabic name and are centered around the most stable name segment in the Arabic name, the GN1.

2.10.3.1. The Arabic keys replace the compressed-name keys produced for Legacy ANA, which have severe limitations for retrieving both predictable and unpredictable variants of the regularized Arabic names.

2.10.3.2. For record adds, all keys will be stored with the source record.

2.10.3.3. Keys will be generated for each SN segment (moved or resident) and for each GN1.

2.10.3.3.1. For record adds, more keys will be generated for HF name segments than for LF segments.

2.10.3.3.2. Search keys will be a combination of SN keys and GN1 keys.

2.10.4. **Function**

2.10.4.1. The AKG will form search keys from a combination of keys for each regularized segment in the Surname field and the regularized GN1.

2.10.4.2. Initials

All names that contain the same first character as the initial will qualify for retrieval on an initial.

2.10.4.3. FNU

All GN1 names qualify for retrieval with a GN1 of FNU (First Name Unknown).

2.10.4.4. Search Keys

2.10.4.4.1. The AKG will generate a set of Search Keys for each input name by conjoining each GN1 key with each SN key of the regularized, repositioned input name.

2.10.4.4.2. All search keys generated for an add will be stored with the record add and associated with the regularized, repositioned form of the name.

2.10.4.5. **Generating Keys**

2.10.4.6. The AKG will produce two categories of keys:

1. Single-Part Key (SI): a key formed from the single name segment (SN or GN1). All Single-Part Keys will be used to form the Search Keys. There are three kinds of Single-Part Key:
   - Primary Key (PK): a key formed on a single name segment (SN or GN1) and used to define the set of keys for that segment;
   - Wild-Card Key (WK): a key based on the Primary Key that contains wild-card characters;
   - Special Key (SP): a key formed on a single name segment and intended to handle specific variation in the regularized name.

2. Search Key (SK): a multipart key that will be stored and used for retrieval, consisting of a combination of the keys associated with every SN segment and those associated with the GN1.

2.10.4.7. **Single-Part Key** (SI)

1. The **Primary Key** (PK)
   - The PK is formed from one name segment.
   - The PK has a maximum of three characters.
   - The PK has the form CCC or CC or C, where C represents any consonant (except in the leftmost position where C may be a vowel).
   - The PK is formed from the leftmost character (vowel or consonant) of the regularized segment and the following two consonants (including H, Y and W). If fewer than two additional consonants are available, then the PK may be shorter.
2. The **Wild-Card Key** (WK)
   - The WK is formed from the Primary Key.
   - The WKs will have the forms *CC, C*C, CC*, where * represents any consonant, except in the leftmost position where it may represent a vowel.
   - The WK will have the forms C* and *C with segments that have only two candidate characters.
   - A WK will ***not*** be formed from a Primary Key with only 1 component (i.e., C).

Figure 4: Example: Formation of Primary and Wild-Card Keys

SEGMENT	PRIMARY KEY	WILD-CARD KEYS		
GAMILA	GML	*ML	G*L	GM*
ABASI	ABS	*BS	A*S	AB*
SAID	SD	*D	S*	
DAI	D	none		

2.10.4.7.1. **Special Key** (SP)

2.10.4.7.2. The AKG will produce Special Keys (SP) to accommodate situations that cannot be accommodate by the ARR.

2.10.4.7.3. The AKG will generate the Special Keys in addition to the Primary and Wild-Card Keys.

2.10.4.7.3.1. **K-Key**

2.10.4.7.3.2. The character K alternates with null in many Arabic names, resulting in the potential overlap of many names with the K names.

2.10.4.7.3.3. This phenomenon is not readily handled by the ARR, so names with a K require a Special Key.

2.10.4.7.3.4. The K-Keys are formed in the following way:

2.10.4.7.3.4.1. For any segment with K in initial position, the following keys are produced: *CC where * represents any character or nothing. (This key is equivalent to a WK produced for this name.)

2.10.4.7.3.4.2. For any segment with K in medial position, the following keys are produced:

1. CkC, where k represents the character "k";
2. CCC, where k has been deleted from the name string and the CCC represents the three leftmost consonants that remain; and

2.10.4.7.3.4.3. For any segment with K in final position, the following keys are produced:

1. CCk, where k represents the character "k" and
2. CC, where k has been deleted from the name string and the CC represents the two leftmost consonants (or vowel in first position) that remain.

2.10.4.7.3.5. The standard set of Wild-Card Keys will also be produced from the Primary Key for K-names.

2.10.4.7.3.6. **Record Add/Query**: The AKG will generate and store all K-Keys with the segment.

Figure 5: Example: Formation of K-Keys

NAME SEGMENT / VARIANT	PRIMARY KEY	K - KEYS	WILD-CARD KEYS
KARSCH	KRS		*RS, K*S, KR*
ARSCH	ARS		*RS, A*S, AR*
MUKBEL	MKB	MBL	*KB, M*B, MK*
MUBEL	MBL		*BL, M*L, MB*
FARUK	FRK	FR	*RK, F*K, FR*
FARU	FR	FR	*R, F*

2.10.4.7.3.7. **High Frequency Key** (HK)

2.10.4.7.3.8. The AKG will generate Special Keys for High Frequency segments found in the input name.

2.10.4.7.3.9. The AKG will access the Arabic Name Type (ANT) Data Store to identify HF segments. (See Section 3.5.)

2.10.4.7.3.9.1. The ANT will contain a set of Arabic name types, the most frequently occurring of which will be specified as High Frequency name segments (HI_FREQ = 1 (True)). (See Section 3.5 for details).

2.10.4.7.3.9.2. The AKG will tag as HF all name segments in the input record that match one of the ARABIC_NAME_TYPE segments for which HI_FREQ = 1 (is True).

2.10.4.7.3.9.3. The AKG will tag all other name segments as LF.

2.10.4.7.3.10. **Record Add/Query**

2.10.4.7.3.11. The AKG will generate and store the Primary Key for any segment that has been tagged as a HF name segment.

2.10.4.7.3.12. **Record Add**

2.10.4.7.3.13. The AKG will generate and store all appropriate Wild-Card Keys for any segment that has been tagged as a HF name segment.

Figure 6: Example: Primary Key as HF Key

HF SEGMENT	PRIMARY KEY	WILD-CARD KEYS
MUHAMAD	MHM	*HM, M*M, MH*
AHMED	AHM	*HM, A*M, AH*
ALI	AL	*L, A*

2.10.4.7.4. **Search Keys (SK)**

2.10.4.7.5. The Search Key is a multipart key formed from all keys associated with one SN segment and all keys associated with the GN1: e.g., *CC + *CC, *CC + C*C, C*C + CC*, etc.

2.10.4.7.6. The Search Keys will be the keys used for retrieval of records from the database.

2.10.4.7.7. **Search Key Formation**

2.10.4.7.8. To form the set of search keys that will be related to each input record, the AKG will combine each SN segment with the GN1 segment: SN1 + GN1, SN2 + GN1, etc.

2.10.4.7.9. The AKG will determine the frequency (HF or LF) of each of the conjoined segments.

2.10.4.7.10. The number and type of Search Keys will be based on the frequency of the name segments.

2.10.4.7.11. The AKG will form

- Standard Search Keys and
- HF Search Keys.

2.10.4.7.12. **Standard Search Keys (SS)**

2.10.4.7.13. Standard Search Keys (SS) are formed for each SN and GN1 pair.

2.10.4.7.14. **Record Add**

2.10.4.7.15. To form a set of Standard Search Keys, the AKG will combine each Wild-Card Key and each K-Key of each SN segment with each Wild-Card Key and each K-Key of the GN1.

2.10.4.7.15.1. For example, each segment with three characters (CCC) will have generated three Wild-Card Keys.

2.10.4.7.15.2. When the keys from two segments with three characters each are paired, there will be a total of nine keys.

2.10.4.7.15.3. For segments with fewer characters, there will be fewer than nine keys.

2.10.4.7.16. The AKG will generate and store these keys with the record.

Figure 7: Example: Formation of Standard Search Keys (Record Add)

REPOSITIONED, REGULARIZED INPUT FORMAT: AHMED BADAWI, MUHAMAD		
	GN1: MUHAMAD	STANDARD SEARCH KEYS
SN1: AHMED	SN1+GN1: AHMED MUHAMAD	*HM*HM, *HMM*M, *HMMH*, A*M*HM, A*MM*M, A*MMH*, AH**HM, AH*M*M, AH*MH*
SN2: BADAWI	SN2+GN1: BADAWI MUHAMAD	*DW*HM, *DWM*M, *DWMH*, B*W*HM, B*WM*M, B*WMH*, BD**HM, BD*M*M, BD*MH*

2.10.4.7.17. **Query**

2.10.4.7.18. If either segment of the SN + GN1 pair has been tagged as LF, the AKG will generate the Standard Search Keys.

2.10.4.7.19. To form a set of Standard Search Keys, the AKG will combine each Wild-Card Key and K-Key of each SN segment with each Wild-Card Key and K-Key of the GN1. (See Section 2.10.4.7.15)

2.10.4.7.20. **HF Search Keys (HS)**

2.10.4.7.21. **Query**

2.10.4.7.22. If both segments (the SN and the GN1) of the conjoined pair have been tagged as HF segments, the AKG will form *one* Search

Key from the Primary Key of the SN + the Primary Key of the GN1.

2.10.4.7.23. The High Frequency Search Key will be the ***only*** Search Key used for a ***query*** on the SN + GN1 pair when both segments are HF segments.

2.10.4.7.24. **Record Add**

2.10.4.7.25. If both segments (the SN and the GN1) that have been conjoined have been tagged as HF segments, the AKG will form *one* Search Key from the Primary Key of the SN + the Primary Key of the GN1.

2.10.4.7.26. The HS will be stored with a record add.

2.10.4.7.27. The HS will be a key stored in addition to the Standard Search Keys for the record.

Figure 8: Example: HF Search Keys

REPOSITIONED, REGULARIZED INPUT FORMAT: AHMED ALI, MUHAMAD		
	GN1: MUHAMAD (HF)	**HF SEARCH KEYS**
SN1: AHMED (HF)	SN1+GN1: AHMED MUHAMAD	AHMMHM
SN2: ALI (HF)	SN2+GN1: ALI MUHAMAD	ALMHM

2.11. Retrieval Function of the Arabic Search Engine (ASE)

2.11.1. The Arabic Search Engine (ASE) will retrieve records from the database based on the following criteria:

- An exact match of the query Search Keys and stored Search Keys and
- Refusal Code Level and associated Year-of-Birth Range.

2.11.2. The ASE will access the Refusal Code Level/Year-of-Birth Range (RLYOB) Data Store to determine the YOB range within each Refusal Level to search for candidate records.

2.11.3. The ASE will retrieve the unique ID and the **regularized, repositioned** form of the record.

2.11.3.1. Determination of the proximity by the Arabic Filter and Sorter of the query and database records will be based on the regularized, repositioned form of the record.

2.11.3.2. The ASE will eliminate all records with the same unique ID retrieved during the retrieval process.

2.11.4. **Subordinates**

Arabic Key Generator.

2.12. ARABIC FILTER AND SORTER MODULE DECOMPOSITION (AFS)

2.12.1. **Identification**

This module is known as the Arabic Filter and Sorter (AFS).

2.12.2. **Type**

2.12.2.1. The AFS is a module that accepts each regularized database record retrieved by the ASE and compares it to the regularized form of the query record.

2.12.2.2. The AFS is constituted of two subordinate functions:

- the Arabic Filter and
- the Arabic Sorter.

2.12.2.3. The AFS must follow the Arabic Search Engine (ASE).

2.12.3. **Purpose**

2.12.3.1. The set of database records that the ASE will retrieve will have no value relative to the query record. The AFS will evaluate each of the records retrieved for its proximity to a query record, will retain those that pass a pre-established threshold and will sort the resultant candidate list.

2.12.3.2. The filtering process will take into account a number of factors that play a role in determining the relative value of Arabic names.

2.12.4. **Function**

2.12.4.1. The AFS will compare the query name and record name to determine a relative surname value and given name value and will generate a composite score for the records by accounting for Date-of-Birth, Refusal Level and Country-of-Birth proximity.

2.12.4.2. **Arabic Filter Function of the AFS**

2.12.4.3. The Arabic Filter and Sorter will first determine if the query record and prime database record (unregularized version) match exactly.

2.12.4.3.1. The Surname, Given Name, Date-of-Birth and Country-of-Birth must be exact matches.

2.12.4.3.2. If the two records match exactly, the AFS will tag the record as an exact match.

2.12.4.3.3. The AFS will send the record directly to the Arabic Sorter Function.

2.12.4.4. The Arabic Filter and Sorter (AFS) will accept the regularized, repositioned candidate records retrieved by the ASE.

2.12.4.5. The AFS will perform a digraph comparison of the regularized, repositioned surname segments (stems) of the query record and each candidate record.

2.12.4.6. The AFS will perform a digraph comparison of the regularized given name segment (stem) of the query record (GN1) and the given name segment (stem) of each candidate record (GN1).

2.12.4.6.1. The score produced by the digraph comparison (DI_VAL) will be adjusted by values assigned to several parameters.

2.12.4.6.2. The score assigned to the surname and to the given name, after the parameters have adjusted the DI_VAL, will be the SN_VAL and the GN_VAL.

2.12.4.6.3. Factors that contribute to the determination of the name scores (SN_VAL and GN_VAL) include

- SNTHR
- GNTHR
- OPVAL
- INITSN
- INITGN
- TAQASN
- TAQAGN
- TAQXSN
- TAQXGN
- GNDR

2.12.4.6.4. A final name score will be calculated for each candidate database record as it compares to the query record.

2.12.4.6.4.1. A score for the SN will be calculated: SN_VAL.

2.12.4.6.4.2. A score for the GN will be calculated: GN_VAL.

2.12.4.6.5. To be included in the final candidate list, the SN_VAL and GN_VAL must each pass pre-determined SN and GN threshold levels (SNTHR and GNTHR).

2.12.4.7. **Surname Evaluation**

2.12.4.8. The AFS will perform a digraph comparison of each SN stem of the database record with each SN stem of the query record.

2.12.4.8.1. The digraph value is determined in the following way:

2.12.4.8.1.1. The digraphs are identified for each name stem.

2.12.4.8.1.1.1. Each pair of alphabetic characters is identified: TAFIQ → TA / AF / FI / IQ

2.12.4.8.1.1.2. A digraph is also formed of the initial boundary (#) and the first alphabetic character: TAFIQ → #T.

2.12.4.8.1.1.3. A digraph is also formed of the final alphabetic character and the final boundary (#): TAFIQ → Q#.

2.12.4.8.1.2. The number of shared digraphs is calculated.

2.12.4.8.1.2.1. A digraph may match one digraph only.

2.12.4.8.1.3. The number of shared digraphs is multiplied by 2 and divided by the total number of digraphs in Comparand #1 added to the total number of digraphs in Comparand #2.

2.12.4.8.1.3.1. The formula is:

$2 * d / a + b$,
where d = the total number of shared digraphs;
where a = the total number of digraphs in Comparand #1; and
where b = the total number of digraphs in Comparand #2.

2.12.4.8.1.4. The result is the Digraph Value (DI_VAL) for the two Comparands.

Figure 9: Example: Digraph Calculation

COMPARANDS	DIGRAPHS	SHARED DIGRAPHS	DI_VAL
COMPARAND #1: BADIR	#B BA AD DI IR R# (6 total digraphs = a)	BA AD DI IR R#	2*d / a + b = 10 / 13
COMPARAND #2: ABADIR	#A AB BA AD DI IR R# (7 total digraphs = b)	= 5 (d)	**0.77**

2.12.4.9. This process is performed for each of pair of Comparands in the database and query SN (SN1/SN1, SN1/SN2, SN1/SN3, SN2/SN2, etc.).

2.12.4.10. Each DI_VAL is adjusted according to parameter values in the Filter Parameter Data Store (see Section 3.6 for details).

2.12.4.11. The AFS will determine if the appropriate parameter conditions are met.

2.12.4.12. If the appropriate conditions are present, the DI_VAL will be multiplied by the value assigned to the parameter and the relative score of the two Comparands will be lowered.

2.12.4.13. **Parameter Conditions**

2.12.4.13.1. **INITSN**: Surname Initial

2.12.4.13.1.1. Definition: A SN segment is a single character and it matches the first character of the other comparand.

2.12.4.13.1.2. Action: Assign the INITSN value to the comparison value (i.e., do not calculate the DI_VAL).

2.12.4.13.2. **OPSN**: Out-of-Place Surname

2.12.4.13.2.1. Definition: A SN segment that is not in the same relative position in the SN string in both the database and query records.

2.12.4.13.2.2. Action: Multiply the DI_VAL by the OPSN value. (See Figures 10 and 11.)

2.12.4.13.3. **TAQ Filter**

2.12.4.13.4. All TAQ tags (ID_NO, disposition, TAQ_TYPE and associated SN stem) will be retrieved with the database record.

2.12.4.13.5. The AFS will evaluate any TAQs associated with the SN segments being evaluated, except Stranded Affixes (see Section 2.5.4.2.7.3).

2.12.4.13.5.1. A Stranded Affix will not play a role in the prefix comparison.

2.12.4.13.6. **Single TAQs**

2.12.4.13.7. **Missing TAQs**

2.12.4.13.8. TAQASN: Absent TAQ Value

2.12.4.13.8.1. Definition 1: One of the two comparands has a TAQ tag, the other does not.

2.12.4.13.8.2. Definition 2: Both SN segments have a single TAQ tag, one is a TAQ DELETE, the other a TAQ DISREGARD.

2.12.4.13.8.3. Action: Multiply the DI_VAL by the TAQASN value. (See Figures 12 and 22.)

2.12.4.13.9. **TAQ DELETE**

2.12.4.13.9.1. If the TAQ DELETE tags refer to the same TAQ segment, the DI_VAL will be unchanged.

2.12.4.13.9.2. If the TAQ DELETE tags refer to different TAQ DELETE segments, multiply the DI_VAL by the TAQXSN value. (See Figure 22.)

2.12.4.13.10. **TAQ DISREGARD Processing**

2.12.4.13.10.1. The AFS will access the TAQ Filter Data Store (TF) to process SN TAQ segments that have been tagged as DISREGARD.

2.12.4.13.10.2. Definition: The AFS will access the TAQ Filter Data Store (TF) to process records if they both contain SN TAQ segments that have been tagged as DISREGARD.

2.12.4.13.10.3. Action 1: The AFS will assign TAQDIS#1 to the TAQ DISREGARD segment for the database SN segment and TAQDIS#2 to the TAQ DISREGARD segment for the query SN segment.

2.12.4.13.10.4. Action 2: If the two TAQ DISREGARD segments match, the DI_VAL will remain unchanged.

2.12.4.13.10.5. Action 3: If the two TAQ DISREGARD segments do not match, the AFS will identify the TF_VALUE for the pair in the TF. (See Figure 24.)

2.12.4.13.10.5.1. The AFS will multiply the DI_VAL by the TF_VALUE for the pair.

2.12.4.13.11. **Multipart TAQs**

2.12.4.13.11.1. Definition: If at least one SN comparand has multipart TAQ tags (they may be all DISREGARD, all DELETE, or mixed DISREGARD/DELETE), the AFS will perform the following analyses.

2.12.4.13.11.2. Action: If all TAQs match, AFS will make no change in the DI_VAL.

2.12.4.13.11.3. **TAQ DELETEs**

2.12.4.13.11.3.1. Definition: All DELETE tags

2.12.4.13.11.3.2. Action 1: If any DELETE TAQ matches, the AFS applies no change.

2.12.4.13.11.3.3. Action 2: If no DELETE TAQs match, multiply the DI_VAL by the TAQXSN Value.

2.12.4.13.11.4. **TAQ DISREGARDs**

2.12.4.13.11.4.1. Definition: All DISREGARD tags

2.12.4.13.11.4.2. Action 1: If any TAQ DISREGARD segment matches, the AFS will make no change in the DI_VAL.

2.12.4.13.11.4.3. Action 2: If no TAQ DISREGARD segments match, the AFS will identify the highest match value from the TF (TF_VALUE) and multiply that by the DI_VAL. (See Figures 23 and 24.)

2.12.4.13.11.5. **TAQ DISREGARD and DELETEs**

2.12.4.13.11.5.1. Definition: Mixed DISREGARD/DELETE tags

2.12.4.13.11.5.2. Action 1: If DISREGARD segments are present in both comparands and there is any match among the DISREGARD segments, the AFS will make no change in the DI_VAL.

2.12.4.13.11.5.3. Action 2: If DISREGARD segments are present in both comparands and there is no match among the DISREGARD segments, the AFS will determine the highest match value from the TF for any DISREGARD tags and multiply the DI_VAL by that value. (That is, ignore any DELETE tags.)

2.12.4.13.11.5.4. Action 3: If a DISREGARD segment is in one comparand and not the other and the two comparands have at least one DELETE tag that matches, the AFS will make no change in the DI_VAL.

2.12.4.13.11.5.5. Action 4: If a DISREGARD segment is in one comparand and not the other and the two comparands have DELETE tags that do not match, multiply the DI_VAL by the TAQXSN. (See Figure 22.)

2.12.4.14. After all evaluations have been performed, the AFS will choose the highest score for each name segment.

2.12.4.14.1. The highest score for **both** the row and column must be chosen.

2.12.4.14.2. Only one score per row and column is permitted.

2.12.4.14.3. If two scores are equal, only one is chosen.

Figure 10: Example 1: Digraph Evaluation: Equal Number of SN Segments; Digraph Variants BADAWI/BEDAWI

	AHMED	ALI	BADAWI
AHMED	**1.00**	0.20	0.00
ALI	0.20	**1.00**	0.18
BEDAWI	0.15	0.18	**0.71**

Figure 11: Example 2: Digraph Evaluation: Different Number of SN Segments; OPSN applies to BADAWI/BEDAWI

	AHMED	ALI	BADAWI
AHMED	**1.00**	0.20	0.00
BEDAWI	0.15	0.18	**0.61**

Figure 12: Example 3: Digraph Evaluation: Same Number of SN Segments; TAQ Tag Present on One SN

	(ABU) AHMED	SALIM	SAYED
AHMED	**0.90**	0.00	0.28
SAID	0.00	0.36	**0.47**
AKBAR	0.16	**0.00**	0.00

Figure 13: Example 4: Digraph Evaluation: Same Number of SN Segments; Different TAQ_DISREGARD Segments Present

	(ABU) AHMED	SALIM	SAYED
(BIN) AHMED	**0.50**	0.00	0.28
SAID	0.00	0.36	**0.47**
AKBAR	0.16	**0.00**	0.00

2.12.4.15. The AFS will sum the values chosen from the comparison matrix and will divide by the number of values chosen to produce the SN_VAL.

2.12.4.15.1. In Example 1, 1.00 + 1.00 + 0.61/3 = 0.87

2.12.4.15.2. In Example 2, 1.00 + 0.61/2 = 0.81

2.12.4.15.3. In Example 3, 0.90 + 0.47 + 0.00/3 = 0.46

2.12.4.15.4. In Example 4, 0.50 + 0.47 + 0.00/3 = 0.32

2.12.4.16. The AFS will compare the SN_VAL to the SNTHR.

2.12.4.16.1. The SN_VAL must be equal to or greater than the SNTHR.

2.12.4.16.2. The record must pass the SNTHR to qualify for the final candidate list.

2.12.4.17. **Given Name Evaluation**

2.12.4.18. The GN has only one segment, the GN1.

2.12.4.18.1. The AFS will perform a digraph comparison on the regularized GN1 stem of the database record and the regularized GN1 of the query record.

2.12.4.18.2. The DI_VAL will be calculated as it was for the SN (see Section 2.12.4.8).

2.12.4.18.3. The DI_VAL will be adjusted by several GN parameters.

2.12.4.18.4. **INITGN**: Given Name Initial

2.12.4.18.4.1. Definition: A GN1 is a single character and matches the first character of the GN1 of the other comparand.

2.12.4.18.4.2. Action: Assign the INITGN value to the comparison value (i.e., do not calculate a DI_VAL)

2.12.4.18.5. **TAQ Evaluation** will proceed as with the SN, mutatis mutandi (see Section 2.12.4.13.3).

2.12.4.18.6. **GNDR**: Record Gender Value

2.12.4.18.6.1. The AFS will compare the record gender of the input name and the query name.

2.12.4.18.6.2. If the genders match, no action will take place.

2.12.4.18.6.3. If the genders do *not* match, multiply the DI_VAL of the GN1 by the GNDR value. (See Figure 24.)

2.12.4.19. The value resulting from all GN1 calculations will be the GN_VAL.

2.12.4.20. The AFS will compare the GN_VAL to the GNTHR. (See Figure 24.)

2.12.4.20.1. The GN_VAL must be equal to or greater than the GNTHR.

2.12.4.20.2. The record must pass the GNTHR to qualify for the final candidate list.

2.12.4.21. **Composite Score**

2.12.4.22. The AFS will develop a Composite Score for the two comparands.

2.12.4.23. The AFS will adjust the GN_VAL and the SN_VAL by factors that reflect the proximity of the Refusal Level, Date of Birth and Country of Birth.

2.12.4.24. The GN_VAL and SN_VAL will be multiplied by factors that apply to the RL, DOB and COB.

2.12.4.25. **Refusal Level Factor**

2.12.4.26. The AFS will access the Refusal Code Level Data Store to determine the Refusal Level Category of the Refusal Code.

2.12.4.27. The AFS will access the Filter Parameter Data Store to find the PARM_VAL associated with the Refusal Level (RL#).

2.12.4.28. **Date-of-Birth Factor**

2.12.4.29. The AFS will access the Year-of-Birth Range Data Store to determine the YOB Category, YOB#, of the Dates-of-Birth of the comparands. The highest value is applied to the relationship.

2.12.4.30. The AFS will access the Filter Parameter Data Store to find the PARM_VAL associated with the YOB Category (YOB#).

2.12.4.31. **Country-of-Birth Factor**

2.12.4.32. The AFS will access the Country of Birth Category Data Store to determine the COB Category, COB#.

2.12.4.33. The AFS will access the Filter Parameter Data Store to find the PARM_VAL associated with the Country of Birth Category (COB#).

2.12.4.34. The AFS will calculate a composite score by multiplying the SN_VAL by the GN_VAL by the RL# PARM_VAL by the YOB# PARM_VAL by the COB# PARM_VAL.

2.12.4.35. **Final Sort Function of the AFS**

2.12.4.36. The AFS will rank order the final candidate list of database records.

2.12.4.37. The prime (unregularized) record will be returned to the user.

2.12.4.37.1. There may be significant differences between the query record and the qualifying database records.

2.12.4.37.2. The Composite Score will be returned with the record.

2.12.4.38. Any record that is tagged as an exact match will be placed at the top of the list.

2.12.4.39. All remaining records in descending order of Composite Score.

2.12.4.40. The goal of the final sort is to place exact record matches on the top and to rank order the remaining records by the degree of contribution that each data element (SN, GN, DOB, COB, Refusal Code Level (RL)) makes to the overall record value.

2.12.4.41. The details of the sort will be derived from extensive discussion about the business requirements.

2.12.4.42. Because the scores from the various pipes may not have been calculated in the same way, a method for evaluating the relative value of candidate records will have to be devised.

2.12.4.43. **Internal Order**

2.12.4.43.1. There may be cases in which the sorting criteria are met equally by more than 1 record.

2.12.4.43.2. Where multiple records qualify equally, there will be an internal sort order.

2.12.4.43.2.1. SN Score

2.12.4.43.2.2. GN Score

2.12.4.43.2.3. DOB Levels

2.12.4.43.2.4. Refusal Levels

2.12.4.43.2.5. COB Relationships

2.12.4.44. The AFS will return the top n records to the central CLASS-E sorter.

2.12.4.44.1. The number of records to be returned will be a system setting.

2.13. LINGUISTIC TRACE FACILITY MODULE DECOMPOSITION

2.13.1. **Identification**

This module is known as the Linguistic Trace Facility (LTF).

2.13.2. **Type**

The LTF is a program that will interact with any or all modules and functions within those modules.

2.13.3. **Purpose**

The LTF will allow system evaluators to access information about the system functions so that the quality of the content can be ensured. To diagnose and remedy problems associated with questionable system results, evaluators must

have access to the results of system functionality at various points during the processing cycle.

2.13.4. **Function**

2.13.4.1. The LTF will be a mechanism that will copy and divert statistics, information, processing results to a file outside the main processing module.

2.13.4.2. The file will be readily accessible on-line for examining by a system evaluator.

2.13.4.3. Multiple trace points will be identified when the system is built.

2.13.4.4. Examples of trace points:

- What ARRs (by ID_NO) have applied
- Regularized, repositioned name form
- All keys generated for a query and for an add
- SN and GN DI_VAL
- SN_VAL and GN_VAL
- Record Gender
- Sort considerations

3. DATA DECOMPOSITION

3.1. DATA

3.1.1. The input data for an ANA-E query will contain all information that is currently required by CLASS and in the standard format required by CLASS.

- NAME (Surname, Given Name);
- DOB (Date of Birth; Day Month Year); and
- COB (Country of Birth; FIPS codes).

In addition, the following will be specified:

- Applicant Gender (AG): Male (M), Female (F), Unknown (U).
- A unique identifier (UID) (as defined in CLASS-E).

3.1.2. For adds, other record information will be entered, as required by CLASS and CLASS-E: e.g., refusal code, province of birth.

3.2. DATA STORES

The following data stores will be accessed by the ANA-E processing components:

- Arabic Regularization Rules Data Store (ARR)
- Arabic Title/Affix/Qualifier Data Store (ATD)
- Arabic Name Type Data Store (ANT)
- Filter Parameter Data Store (FP)
- TAQ Filter Data Store (TF)
- Refusal Code Level Data Store (RCL)

- YOB Range Data Store (YR)
- Refusal Code Level/Year-of-Birth Range Data Store (RLYOB)
- COBPROX Data Store (COBPROX)
- Arabic COB Category Data Store (ACOB)

3.3. ARABIC REGULARIZATION RULES DATA STORE DECOMPOSITION

3.3.1. **Identification**

This rule base is known as the Arabic Regularization Rule Base (ARR).

3.3.2. **Type**

3.3.2.1. The ARR is a set of transformation rules accessed by the Arabic Rule Engine.

3.3.2.2. The ARR will have the following format:

Figure 14: Format: Arabic Regularization Rule Base

FIELD NAME	DATA TYPE	FIELD SIZE	DATA VALUE
ID_NO	integer	3	001...999
PRE-CONTEXT	character	unlimited	any ASCII character
IN	character	unlimited	any ASCII character
POST-CONTEXT	character	unlimited	any ASCII character
OUT	character	unlimited	any ASCII character

3.3.2.3. Definitions

3.3.2.3.1. ID_NO: a unique, arbitrary numerical reference to the rule.

3.3.2.3.2. PRE-CONTEXT: preceding context for the element to be matched; delimited by preceding and following quotation marks (“ ”)

3.3.2.3.3. IN: the match context; the portion of the name that will undergo change; delimited by preceding and following quotation marks (“ ”)

3.3.2.3.4. POST-CONTEXT: trailing context for the element to be matched; delimited by preceding and following quotation marks (“ ”)

3.3.2.3.5. OUT: the rule output; the realized change in the IN; delimited by preceding and following quotation marks (“ ”)

3.3.2.4. There is no internal limit on the size of the Pre-Context, In, Post-Context or Out, although the system may have an external limit (e.g., the maximum size of the SN field).

3.3.2.5. All rules will use standard regular expression notation, with one exception ($), which has been defined specifically for this rule base.

3.3.2.6. **Regular Expression Notation**

Figure 15: Regular Expression Notation

REGEXP NOTATION	DEFINITION
.	Matches any single character, including white space.
-	Stands for all characters that come between the two characters given. This is a standard "from-to" notation; with characters, it presumes an A to Z character set. For example, [A-D] will match on A or B or C or D. (See [ ] below.)
[ ]	Identifies a class of characters; a match can occur on a single occurrence of any single element within the [ ]: [OU] will match O or U. For example, "J[OU]N" will match on JON or JUN but not on JOUN. If a + is added to the bracketed expression, "J[OU]+N", it will match on any combination of any number of Os and Us: JOUUN, JUUON, JOUOUN, JUUN, JOUUN, etc. In contrast, OU without [ ] will match only on the exact combination of characters OU: "JOUN" will match on JOUN only.
+	Matches one or more occurrences of a preceding character or regular expression, in any order. For example, JO+N will match JON or JOON or JOOON, etc., [OU]+ matches OOOU or OUOUOU or O or UO or OOU or UUUO, etc.
?	Matches zero or ***one*** occurrence of the preceding regular expression. The expression "JOH?N" will match on JON or JOHN but will not match on JOHHN.
*	Matches zero or more occurrences of the preceding regular expression. The expression "JOH*N" will match on JON, JOHN, JOHHN, JOHHHHN, etc.
( ) .	Groups together regular expressions. "(J[OU]N|H[AE]RRY)" will match on JON or JUN or HARRY or HERRY. (Contrast with [ ] which identifies a character class and contrast with { } which identifies a metasymbol.)
| → (|)	Matches ***either*** the preceding regular expression ***or*** the following regular expression. The full expression is all contained within ( ). For example, (AB|AP) will match the character string AB ***or*** AP. The same expression may also be written A[BP].
" "	Defines the context boundary: the Preceding Context, Match Context, Post Context and Output. A context that is made up of only **one** metasymbol and is not bracketed by { } should not be surrounded by " ". For example, the metasymbol Consonant can stand alone. If the metasymbol is enclosed within {}, then all regular expressions contained within the context must be enclosed within " ". For example, "{Consonant}{Letter}" within one context requires both { } and " " .
{ }	Contains one or more pre-defined metasymbols. If { } are used, they must be surrounded by " ". If a single metasymbol occurs alone, no { } are necessary and therefore no " " are necessary. For example, the single metasymbol Vowel can appear either as Vowel or "{Vowel}". If more than one element is used, the metasymbol must all appear within { } and the whole string within " ". E.g., Vowel, "{Vowel}", "J{Vowel}HN" are acceptable formats.
$	Indicates the character to output. $ is defined differently from other standard definitions. $ is a variable that is followed by an integer that references a character in the match string. For example, each character in an input string is associated with a different, consecutive integer value, up to the number of characters in the match: JONES becomes J = $1, O = $2, N = $3, E = $4, S = $5. Reference can be made to the index values in the output string. $1 $2 $2 $3 $3 $5 would represent JOONNS.

3.3.2.7. **Metasymbols**

A number of meta-symbols will be accessed by the rules. The metasymbols are variables declared at the beginning of the ARR Data Store.

Figure 16: ARR Metasymbols

METASYMBOL	DEFINITION
Letter	"[A-Z]"
Consonant	"[BCDFGHJKLMNPQRSTVWXYZ]" (N.B. Includes W and Y)
Nog (= No Glide)	"[BCDFGHJKLMNPQRSTXZ]" (N.B. No W or Y)
Alla	"[AEOUI]+L+[EA]+H?"
Rhyme	"[AEIOUY]+[BCDFGHJKLMNPQRSTVWXYZ]"
Kesra	"([EA]?[IY]+|[IE]+|Y)"
Dad	"(Z+|TH+|DH+|DD+)"
Tha	"(Z+|TH+|DH+|C+|S+|T+)"
Jim	"(DJ|J|Y|G|DZH|DSCH|GG|DY)"
Gine	"(G|GH|RH)"
Qaf	"(Q|G|K|J|KH|GH|C|QU|CK)"
Kha	"(KH|K|X|Q|C)"
Marbuta	"([EAI]H|[AE]T?)"
Sun	"(C|S|N|D|T|R|Z|G|J)"
Sungem	"(SS|NN|DD|TT|RR|ZZ|GG|JJ)"
Moongem	"(BB|FF|GG|HH|JJ|KK|MM|NN|PP|QQ|VV|WW|XX)"
Vowelgem	"(AA|EE|II|OO|UU)"
Vowel	"[AEIOU]"
Didi	"(KHKH|SHSH|GHGH|RHRH|DHDH|THTH|CHCH|PHPH)"
Dig	"[KSGRDTCP]H"
Bound	" " (= white space)
Anything	"."
Othergem	"(CC|LL|YY)"

3.3.2.8. **Purpose**

The ARR allow records with highly divergent spellings and/or representations of the same name to be retrieved from the database. Usual character comparison techniques are unable to retrieve records with these variants.

3.3.2.9. **Function**

The ARR applies relevant rules to each Arabic name field and produces a common representation for variant realizations of the same name. **MUHAMMED, MOHAMMAD** and **IMHEMED** are variant forms of the same name; each will be set equal to one single representation of the name: **MUHAMAD**, for example. The successful application of one or more rules will produce as output a regularized Arabic name string.

3.3.2.10. Examples

3.3.2.10.1. Example 1 contains two rules that apply to variants of ABDULLA:

- EVDILLAH
- ABD ALA
- ABDU ALLA
- ABDULLAH
- OABDELA
- AABDILA
- ABDELILA

Figure 17: Example 1: "ABDULLA" Regularization Rules

ID _NO	PRE-CONTEXT	IN (MATCH CONTEXT)	POST-CONTEXT	OUT
676	Bound	"[HKCQ]?[AE]+[BV]*D+{Vowel}?{Bound}{Alla}"	Bound	"abdula"
677	Bound	"[HKCQ]?[AE]+[BV]*D+[AEIOU]*L+[IE]*L*AH?"	Bound	"abdula"

3.3.2.10.2. Example 2 contains one rule that applies to variants of G:

- MAGUID
- MADZHID
- MADSCHID
- MADJID
- MAJID
- GHASSAN

Figure 18: Example 2: "G" Regularization Rules

ID _NO	PRE-CONTEXT	IN (MATCH CONTEXT)	POST-CONTEXT	OUT
132	Anything	"(DJ|GH|DSCH|DZH|J+)"	Anything	"g"

3.4. ARABIC TITLE/AFFIX/QUALIFIER DATA STORE DECOMPOSITION

Because the ANA-E design is viewed as an independent sub-program of the CLASS-E system, the Arabic Title/Affix/Qualifier Data Store is presented here as a separate table. It is strongly suggested, however, that CLASS-E support one TAQ Data Store in which the cultural affinity of each TAQ segment is indicated. This is reduce table maintenance and will provide a global picture of the handling of TAQs.

3.4.1. **Identification**

This data store is known as the Arabic Title/Affix/Qualifier Data Store (ATD).

3.4.2. **Type**

The ATD is a data store that contains the Arabic-specific Title, Affix and Qualifier segments and their distribution. It will be accessed by the Arabic Preprocessor (APP) and the Arabic Filter and Sorter.

Figure 19: Format: Arabic TAQ Data Store

DATA FIELD	DATA TYPE	FIELD SIZE	DATA VALUE
ID_NO	integer	4	1...9999
TAQ FORM	character	15	alphabetics
TAQ TYPE	character	1	T, P, I, S, Q
DELETE	integer	1	1, 0 (True, False)
DISREGARD	integer	1	1, 0 (True, False)

3.4.2.1. Definitions

3.4.2.1.1. ID_NO: a unique, arbitrary number that identifies the TAQ segment.

3.4.2.1.2. TAQ FORM: the string that represents the TAQ; the TAQ FORM may be a multipart string (i.e., a string that includes internal white space).

3.4.2.1.3. TAQ TYPE: an indicator of the kind of TAQ segment present: a title (T), prefix (P), infix (I), suffix (S) or qualifier (Q).

3.4.2.1.4. DELETE:

3.4.2.1.4.1. The segment is to be removed from all further consideration in the name search process; it will contribute marginally to the filtering process. It will be returned with the record to the user.

3.4.2.1.4.2. The segment is referenced in the filtering process.

3.4.2.1.4.3. The segment is not removed from the original record and is returned with the record to the user.

3.4.2.1.4.4. True (1) or False (0) indicates whether or not this function is to apply to the segment(s) under consideration.

3.4.2.1.5. DISREGARD:

3.4.2.1.5.1.The segment is to be removed from further consideration in the name search process but will undergo special evaluation in the filtering process. It will be returned with the record to the user.

3.4.2.1.5.2.True (1) or False (0) indicates whether or not this function is to apply to the segment(s) under consideration.

3.4.3. **Purpose**

Peripheral elements (Titles, Affixes, and Qualifiers) in names do not contribute as much to the name evaluation as does the name stem. Identifying and

removing these elements in the name processing component is important. They do, however, contribute to the overall value of a name when compared to another name. They will therefore contribute some value to the filtering and sorting processes.

3.4.4. **Function**

The ATD serves as a repository for all TAQ values and for the treatment that each will be subjected to.

3.5. ARABIC NAME TYPE DATA STORE DECOMPOSITION

3.5.1. **Identification**

This data store is known as the Arabic Name Type Data Store (ANT).

3.5.2. **Type**

3.5.2.1. The ANT is a data store of unique regularized Arabic name segments.

3.5.2.2. The ANT is generated only after regularization has applied to the input name.

3.5.2.3. The ANT will have the following format:

Figure 20: Format: Arabic Name Type Data Store

DATA FIELD	DATA TYPE	FIELD SIZE	VALUE RANGE
ID_NO	integer	5	00001...99999
ARABIC_NAME_TYPE	character	24	alphabetics
GENDER	character	1	M, F, U
HI_FREQ	integer	1	1, 0 (True or False)

3.5.2.4. Definitions

3.5.2.5. ID_NO: a unique, arbitrary numerical reference to the name segment (ARABIC_NAME_TYPE)

3.5.2.6. ARABIC_NAME_TYPE: unique entries that correspond to the ***regularized*** form of a name segment.

3.5.2.7. GENDER: the gender associated with a particular name segment: M (Male), F (Female), U (Unknown/Unspecified). As records are added to the ANT, gender will be specified as U. The gender assigned to new table entries will be periodically reevaluated so that names that can be identified for gender can be appropriately marked.

3.5.2.8. HI_FREQ: the frequency of all names will be indicated. True (1) will indicate that a name segment is considered a high frequency Arabic name segment. All other segments will be marked as False (0), a low-frequency name segment.

3.5.3. **Purpose**

The purpose of the ANT data store is to reduce the need to perform repeated digraph comparisons on a large store of names and to permit the retrieval of gender-matching records.

3.5.4. **Function**

The ANT will provide information about the distinct Arabic name types, their frequency and gender.

3.6. FILTER PARAMETER DATA STORE DECOMPOSITION

3.6.1. **Identification**

This module is known as the Filter Parameter Data Store (FP).

3.6.2. **Type**

3.6.2.1. The FP is a data store that will be accessed by the Filter Component of the Arabic Filter and Sorter (AFS).

3.6.2.2. The FP is a parameter table that will be accessible to and adjustable by the user and whose cell values will be determined through testing and comparative evaluation.

3.6.2.3. The FP has the following format:

Figure 21: Format: Filter Parameter Data Store

DATA FIELD	DATA TYPE	FIELD SIZE	VALUE RANGE	DATA VALUE
PARM_NAME	character	6	alphabetics	SNTHR, GNTHR, OPSN, INITSN, GNDR, INITGN, TAQASN, TAQAGN, TAQXSN, TAQXGN,

				RL#, YOB#, COB#
PARM_VAL	decimal	4	0.00...1.99	Various (TBD)

Figure 22: Example: Filter Parameter Data Store (Values are for example only.)

PARM_NAME	PARM_VAL
SNTHR	0.60
GNTHR	0.65
OPSN	0.60
INITSN	0.85
INITGN	0.85
GNDR	0.65
TAQASN	0.90
TAQAGN	0.90
TAQXSN	0.85
TAQXGN	0.85
RL0	1.20
RL1	1.15
RL2	1.10
RL3	1.05
RL4	1.00
YOB0	1.30
YOB1	1.25
YOB2	1.20
YOB3	1.15
YOB4	1.10
YOB5	1.05
YOB6	1.00
COB1	1.20
COB2	1.15
COB3	1.10
COB4	1.00
COB5	0.95

3.6.2.4. The values provided are as examples only and do not necessarily represent the PARM_VALs to be used for the parameters.

3.6.3. **Purpose**

The FP is a data store that allows easy access to adjustable parameters that contribute to the determination of the composite score of two record comparands.

3.6.4. **Function**

The FP functions as an independent data store with all the adjustable parameters needed by the AFS during the filtering process.

3.7. TAQ FILTER DATA STORE DECOMPOSITION

3.7.1. **Identification**

This data store is known as the TAQ Filter Data Store (TF).

3.7.2. Type

3.7.2.1. This TF will be accessed by the Arabic Filter and Sorter and provides parameter factors for matching TAQ DISREGARD tags during record filtering.

3.7.2.2. The format of the TF follows:

Figure 23: Format: TF Matrix Design

DATA FIELD	DATA TYPE	FIELD SIZE	VALUE RANGE	DATA VALUE
TAQDIS#1	character	8	alphabetics	TAQ_DISREGARD ITEM
TAQDIS#2	character	8	alphabetics	TAQ_DISREGARD ITEM
TF_VALUE	decimal	4	0.00...1.00	Various (TBD)

3.7.2.3. Definitions

3.7.2.4. TAQDIS#1: is the TAQ DISREGARD segment that occurs in one or the other (different) of the comparands.

3.7.2.5. TAQDIS#2: is the TAQ DISREGARD segment that occurs in one or the other (different) of the comparands.

3.7.2.6. TF_VALUE: is the factor that will be used to adjust the SN_VAL or GN_VAL if the TAQDIS#1 and TAQDIS#2 are present in the comparands.

Figure 24: Example: TF Sample (Values are for example only)

TAQDIS#1	TAQDIS#2	TF_VALUE
ABD EL	ABD EL	1.00
ABD EL	ABU	0.75
ABD EL	AL	0.85
ABD EL	BIN	0.75
ABD EL	EL DIN	0.50
ABU	ABU	1.00
ABU	AL	0.85
ABU	BIN	0.50
ABU	EL DIN	0.85
AL	AL	1.00
AL	BIN	0.85
AL	EL DIN	0.50
BIN	BIN	1.00
BIN	EL DIN	0.85
EL DIN	EL DIN	1.00

3.7.3. Purpose

Arabic names often have peripheral name elements. Some of these make up a segment of the name, the TAQ values identified in the TF. Their relative value, however, varies. Some of them cannot cooccur, some have opposite

meanings, so it is necessary to identify their relative value when they are contrasted with one another.

3.7.4. **Function**

The TF provides the resources for the AFS to determine the relative value of two TAQs that occur in two comparands.

3.8. REFUSAL-CODE LEVEL DATA STORE DECOMPOSITION

3.8.1. **Identification**

This data store is known as the Refusal Code Level Data Store (RCL).

3.8.2. **Type**

3.8.2.1. It is recommended that the RCL be a parameter file, which can be accessed by the client so RC categories can be added to or changed with ease.

3.8.2.2. The RC data store will provide a list of the Refusal Codes and the level of seriousness of each Refusal Code.

3.8.2.3. The RCL has the following format:

Figure 25: Format: Piece of Refusal Code Level Data Store (RCL)

DATA FIELD	DATA TYPE	FIELD SIZE	VALUE	CATEGORY DEFINITION
00	alphanumerics	3	RL0	Most serious RC: 00
23	alphanumerics	3	RL1	Type 1 Serious RCs
6C	alphanumerics	3	RL2	Type 2 Serious RCs
07	alphanumerics	3	RL3	Type 1 Non-serious RCs
G	alphanumerics	3	RL4	Type 2 Non-serious RCs
...				

3.8.2.4. Definitions

3.8.2.4.1. DATA FIELD: indicates each Visa Refusal Code (Codes and their Refusal Level (see VALUE) are for example only; they do not represent the complete list nor the accurate assignment of a Refusal Code to a Refusal Level).

3.8.2.4.2. DATA TYPE: The RL# will appear in the form RL1, RL2, etc.

3.8.2.4.3. VALUE: RL# is the Refusal Level category to which are particular Refusal Code has been assigned. The Visa Office will assign Refusal Codes to one of 4 categories: RL1, RL2, RL3, RL4; RL0 is reserved for the Refusal Code 00. (The current distinction among Refusal Codes is a binary one: serious and non-serious. Assignment of Refusal Codes to more groups has not yet been done; the consequence is that one or more of these categories may not have a distinct value.) The RL# occurs in

ascending order, from most serious to least serious Refusal Code. The RL# will be linked to a Year-of-Birth Code (see Section 3.9) to determine the relevant subsets of records to be searched.

3.8.2.4.4. CATEGORY DEFINITION:

- **RC0** refers to the Refusal Code 00.
- **RC1** refers to all Refusal Codes that have been designated as Type 1 Serious RC 1, i.e., the most serious, excluding 00.
- **RC2** refers to all Refusal Codes that have been designated as Type 2 Serious RC, i.e., serious but less serious than RC0 and RC1.
- **RC3** refers to all Refusal Codes that have been designated as Type 1 Non-Serious RC. These codes are less serious than the RC0, RC1 and RC2 codes.
- **RC4** refers to Refusal Codes that have been designated as Type 2 Non-Serious. These codes are the least serious codes, less serious than the RC0, RC1, RC2 and RC3 codes.

3.8.3. **Purpose**

It has long been desirable to make more granular distinctions among the Refusal Codes. For many years, DOS has maintained a distinction between serious and non-serious codes; these different categories were correlated with different YOB search ranges. However, a mechanism for making greater distinctions will provide greater flexibility in delimiting the set to be retrieved during the first stage of record analysis. The introduction of five refusal code levels also provides the opportunity to correlate more year-of-birth ranges to the refusal code levels.

3.8.4. **Function**

The RCL provides information needed for the evaluation of record proximity in the Arabic filtering process and contributes to the delimitation of database records retrieved through the RL/YOB Data Store.

3.9. YEAR-OF-BIRTH RANGE DATA STORE DECOMPOSITION

3.9.1. **Identification**

This data store is known as the Year-of-Birth Range Data Store (YR).

3.9.2. **Type**

3.9.2.1. It is recommended that the YR be a parameter file, which can be accessed by the client so YOB ranges can be set. Alternatively, it could be represented as a system parameter whose value(s) are set in an .ini file.

3.9.2.2. The YR will define the YOB ranges that will be associated with a Refusal Level (see Section 3.8).

3.9.2.3. This data store has the following format:

Figure 26: Format: Year-of-Birth Range Data Store (YR)

DATA FIELD	DATA TYPE	FIELD SIZE	VALUE	DATA DEFINITION
YOB0	integer	1	0	exact date of birth
YOB1	character	1	A	exact year, inverted month and day
YOB2	character	1	B	exact year of birth
YOB3	integer	2	1...99	narrow year of birth range
YOB4	integer	2	1...99	standard year of birth range
YOB5	integer	2	1...99	wide year of birth range
YOB6	integer	2	1...99	unlimited year of birth range

3.9.2.3.1. Definitions

3.9.2.3.2. DATA FIELD: YOB# is the Year-of-Birth Range category whose value indicates the year-of-birth range to be searched. The year-of-birth VALUE indicates the search range, that is, the number of years on either side of a given year-of-birth to be searched. For example, if the input year is 1962 and YOB3 range is 4, the search will cover a range of nine years, 1958-1966. The range includes the full year, so all of 1958 and all of 1966.

3.9.2.3.2.1. There are seven YOB# categories, YOB0, YOB1, YOB2, YOB3, YOB 4, YOB5, YOB6.

- **YOB0** is a single integer that refers to an exact month, day, year of birth. If YOB0 is specified, the system must be able to match the month, day and year of the Date of Birth of an input record and a database record.
- **YOB1** is a single character (A) that refers to an exact year-of-birth with the month and day inverted.
  1. If YOB1 is specified, the system must be able to match the year of Date of Birth and an inverted month and day (DEC 03 → MAR 12) of the input record and the database record.
  2. YOB1 will be relevant to the Arabic Filter and Sorter, but may not function as a search parameter since the value would be subsumed in YOB2.
- **YOB2** is a single character (B) that refers to an exact year-of-birth. If YOB2 is specified, the system must be able to match the year of the Date of Birth of an input record and a database record.
- **YOB3** is a one- or two-place integer (1...99) that refers to a narrow year-of-birth range. Narrow year-of-birth range is usually defined as 1 year (for a search range of 3 years).

- **YOB4** is one- or two-place integer (1...99) that refers to a standard year-of-birth range. Standard year-of-birth range is usually defined as 3 years (for a search range of 7 years).
- **YOB5** is a one- or two-place integer (1...99) that refers to a wide year-of-birth range. Wide year-of-birth range is usually defined as 5 years (for a search range of 11 years).
- **YOB6** is a one- or two-place integer (1...99) that refers to an unlimited or extremely wide year-of-birth range. Unlimited year-of-birth range would be set sufficiently high to include all (or all desired) years-of-birth in the database (e.g., 50).

3.9.3. **Purpose**

This YR provides a greater granularity in the year-of-birth range and, therefore, greater flexibility in delimiting the set to be retrieved during the first stage of record analysis. The correlation of five refusal code levels to different year-of-birth ranges will help to delimit the number of records to be searched and to define the more valuable set of records.

3.9.4. **Function**

3.9.4.1. The YR permits greater granularity in the Date-of-Birth types related to the system.

3.9.4.2. The YR will be accessed by the Refusal Code Level/YOB Range Data Store, which will limit the retrieval range in the Arabic Search Engine.

3.9.4.3. The YR will contribute to the Arabic Filter and Sorter to contribute information to the composite score.

3.10. REFUSAL CODE LEVEL / YOB RANGE DATA STORE MODULE DECOMPOSITION

3.10.1. Identification

This data store is known as the Refusal Code Level/YOB Range Data Store (RLYOB).

3.10.2. **Type**

3.10.2.1. The RLYOB is a matrix that merges the values in the Refusal Code Level (RCL) Data Store and the Year-of-Birth Range (YR) Data Store.

3.10.2.2. For each Refusal Level (RL), a Year-of-Birth (YOB) Range is specified.

3.10.2.2.1. Only one YOB Range for each RL is permitted.

3.10.2.2.2. The same YOB Range may apply to more than one RL.

3.10.2.3. The RLYOB has the following format:

Figure 27: Format: Refusal Level/Year-of-Birth Range Data Store (RLYOB)

DATA FIELD	DATA TYPE	FIELD SIZE	VALUE RANGE	DATA VALUE
RL#	character	3	RL0...4	RL0, RL1, RL2, RL3, RL4
YOB#	character	4	YOB0...6	YOB0, YOB1, YOB2, YOB3, YOB4, YOB5, YOB6

Figure 28: Example: RLYOB Data Store

RL#	YOB#
RL0	YOB5
RL1	YOB4
RL2	YOB3
RL3	YOB3
RL4	YOB2

3.10.2.4. Definitions:

3.10.2.5. RL#: is a character string that indicates the Refusal Level of the Refusal Code.

3.10.2.6. YOB#: is a character string that indicates the Date-of-Birth Range Category of the comparands.

3.10.3. **Purpose**

Retrieval of records from the database should be delimited by a relationship between the Refusal Code Level and the Year-of-Birth Range. It will restrict the number of records to be reviewed.

3.10.4. **Function**

The RLYOB is a resource for the Arabic Search Engine to delimit the records retrieved from the database.

3.11. COUNTRY-OF-BIRTH PROXIMITY DATA STORE DECOMPOSITION

3.11.1. **Identification**

This data store is known as the Country-of-Birth Proximity Data Store (COBPROX).

3.11.2. **Type**

3.11.2.1. The COBPROX is a matrix whose cells contain a decimal that reflects the degree of relationship between the country represented on the x-axis and the country represented on the y-axis.

3.11.2.2. The COBPROX has the following format:

Figure 29: Format: COBPROX Data Store

DATA FIELD	DATA TYPE	FIELD SIZE	VALUE RANGE	DATA VALUE
COB#1	character	4	alphabetics	COB Code
COB#2	character	4	alphabetics	COB Code
COBVAL	decimal	4	0.00...1.00	Various

Figure 30: Example: Piece of COBPROX Data Store

COB#1	COB#2	COBVAL
AGS	AGS	1.00
AGS	ALG	0.05
AGS	MORO	0.05
AGS	SARB	0.05
AGS	SYR	0.05
ALG	ALG	1.00
ALG	MORO	0.85
ALG	SARB	0.75
ALG	SYR	0.75
MORO	MORO	1.00
MORO	SARB	0.75
MORO	SYR	0.75
SARB	SARB	1..00
SARB	SYR	0.75
SYR	SYR	1.00
....		

3.11.2.3. Definitions:

3.11.2.3.1. COB#1: is the 4-character COB Code of one of the comparands.

3.11.2.3.2. COB#2: is the 4-character COB Code of one of the comparands.

3.11.2.3.3. COBVAL: is the decimal value assigned through the ACOB (and other COB Category Data Stores).

3.11.3. **Purpose**

The COBPROX Data Store provides information on the relative value of the COBs in two comparands. This value can serve to limit the COBs that are accessed for retrieval.

3.11.4. **Function**

The COBPROX is populated by the ACOB and any other partition-specific Country-of-Birth Category Data Stores. The COBPROX provides COB relationship information.

3.12. ARABIC COUNTRY-OF-BIRTH CATEGORY DATA STORE DECOMPOSTION

3.12.1. **Identification**

This data store is known as the Arabic Country-of-Birth Category Data Store (ACOB).

3.12.2. **Type**

This ACOB is a data store that will be serve as the source of information for the COBPROX Data Store, supplying the COBVAL, and will provide the COB Category (COBCAT) necessary for the Arabic Filter and Sorter.

Figure 31: Format: Arabic Country-of-Birth Category Data Store (ACOB)

DATA FIELD	DATA TYPE	FIELD SIZE	VALUE RANGE	DATA VALUE
COB#1	characters	4	alphabetics	COB Code
COB#1	characters	4	alphabetics	COB Code
COBCAT	characters	5	alphanumberics	COB1...COB99
COBVAL	decimal	4	0.00...1.00	Various

3.12.3. Definitions

3.12.3.1. COB#1: is the 4-character COB Code of one of the comparands.

3.12.3.2. COB#2: is the 4-character COB Code of one of the comparands.

3.12.3.3. COBCAT: is the category assigned to the relationship of two COBs.

3.12.3.3.1. Categories might include such relationships as Exact, State, Geographic Region, Dialect Region.

3.12.3.3.2. All relationships are adjustable.

3.12.4. COBVAL: is the decimal value that will be assigned to a particular COB relationship; this value will be used to determine the COBs that will be permitted in the retrieval process.

3.12.5. Example COB Categories might be:

COB1: Exact represents an exact match of the COBs: ALG/ALG; the COBPROXVAL would be 1.00.

COB2: Western Dialect Region represents the set of COBs that are in close geographic proximity and share naming conventions: ALG/MORO. The score would be something less than that applied to an exact match but nonetheless high: 0.85.

COB3: Arabic Partition represents all COBs within the Arabic partition. The value assigned would be less than that for COB2: 0.75.

COB4: All refers to all COBs and is assigned a value that will allow the search of all COBs; it would be the lowest decimal value used.

Figure 32: Example: Piece of ACOB (Values for example only.)

COB#1	COB#2	COBCAT	COBVAL
ALG	ALG	COB1	1.00
ALG	MORO	COB2	0.85
ALG	SARB	COB3	0.75
ALG	SYR	COB3	0.75
MORO	MORO	COB1	1.00
MORO	SARB	COB3	0.75
MORO	SYR	COB3	0.75
SARB	SARB	COB1	1.00
SARB	SYR	COB3	0.75
SYR	SYR	COB1	1.00

3.12.6. **Purpose**

Pre-defined COB category relationships will provide a definition of the values that appear in the COBPROX Data Store.

3.12.7. **Function**

These COB categories will provide information about COB relationships that will contribute to determination of the Composite Score in the Arabic Filter and Sorter.